Appendix for What makes transfer learning work for medical images: feature reuse & other factors

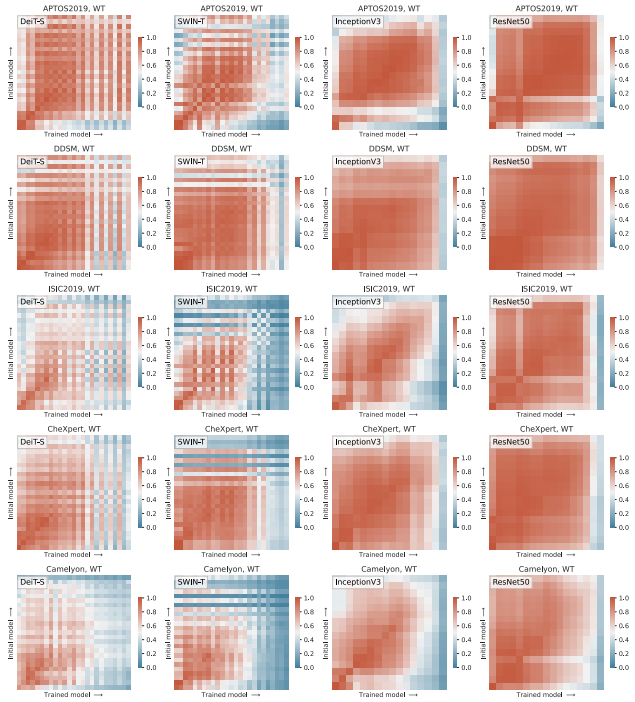

Figure 9. *Feature similarity between the initial and the fine-tuned model for WT initialization.* CKA feature similarity comparison between WT initialized models before and after fine-tuning. Reported for each dataset (rows) and model type (columns). Evidently, models with low inductive bias (fist two columns) indicate strong feature reuse on the early layers. On the other hand, models with strong inductive biases (last two columns) appear a more uniform pattern throughout the network. Note that for all models, more data seem to affect more high-level features.

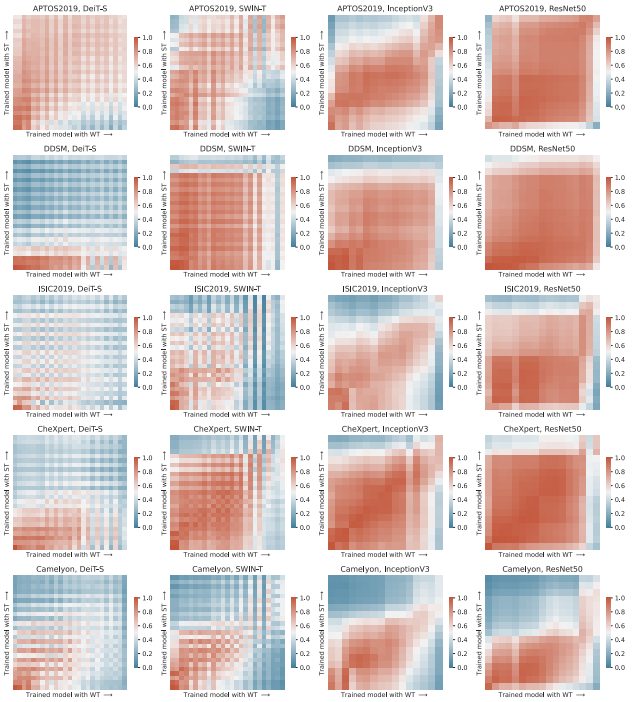

Figure 10. *Feature similarity between WT and ST fine-tuned models.* CKA feature similarity comparison between WT and ST initialized models, calculated at each layer. Reported for each dataset (rows) and model type (columns). The results indicate that for models with low inductive bias, like DEITs, the early layers of ST-initialized networks have increased similarity with early-to-mid layer features from the WT-initialized model, suggesting that the ST-initialized model learns more global features in the early layers. On the other hand, models with strong inductive biases, like RESNET50, seem to learn similar features regardless their initialization, suggesting that inductive biases may somehow naturally lead to similar features.

Appendix overview Here, we provide further details and results from the experiments carried out in this work. Section A provides further experiment details and supplementary figures for the feature similarity experiments. Section B for the layer-wise k-NN evaluation experiments. Section C for the layer-wise re-initialization and Section D for the ℓ_2 weight distance experiments. In Section E we show the mean attended distance for DEIT-S using different initialization schemes. In Sections F and G, we provide additional details for the convergence speed and the architectures we used to evaluate the behaviour of different model capacities. In Section H, we describe in detail the modules used for the WT-ST-n/m initialization schemes and the intermediate layers that we used for the re-initialization, ℓ_2, k-NN and representational similarity experiments. Finally, in Section I we provide additional details for the 5-layer DEIT-S model and we show that it performs comparably with the full 12-layer model.

A. Feature similarity

Details of the feature similarity calculations. The feature similarity throughout this work is measured using Centered Kernel Alignment (CKA). CKA computes the feature

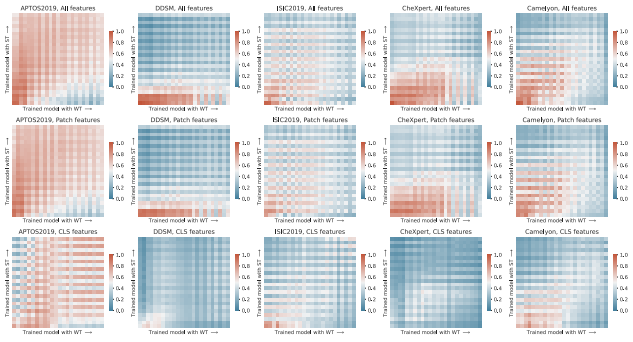

Figure 11. *Feature similarity of WT and ST fine-tuned models for different feature-types.* CKA feature similarity comparison between WT and ST initialized models in DEIT-S, using three different token types (CLS, Patches, CLS + Patches), for each dataset (columns) and embedding type (rows). Evidently, the emergence of global features in the early layers of ST initialized models is associated mainly with the patch tokens (which represent the image patches) whilst the CLS token learns different features depending on its initialization.

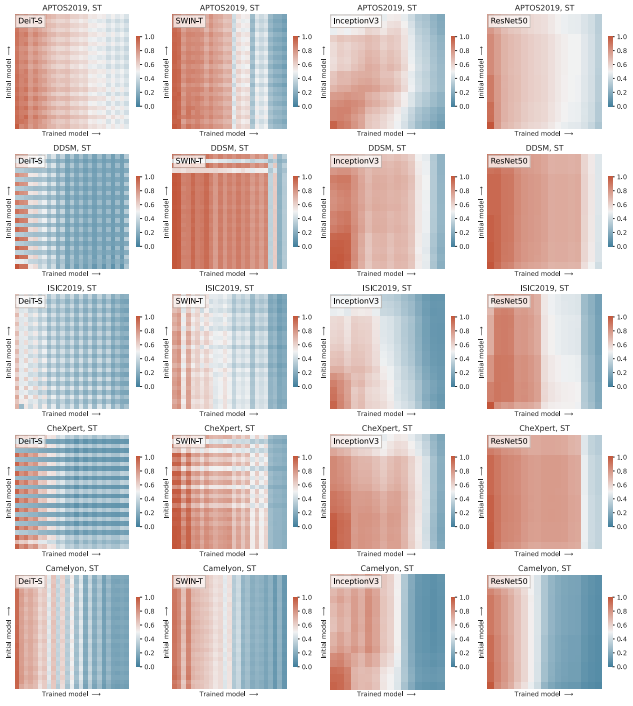

Figure 12. *Feature similarity between initial and fine-tuned model for ST initialization.* CKA feature similarity comparison between ST initialized models before and after fine-tuning. Reported for each dataset (rows) and model type (columns). As we can see, models with no inductive biases exhibit changes throughout the network, while models with increased inductive bias focus more on the mid-to-high layers during training.

similarity between two representations, allowing us to compare those of different layers and/or different models. For a more detailed description of CKA see [34] and [31]. The

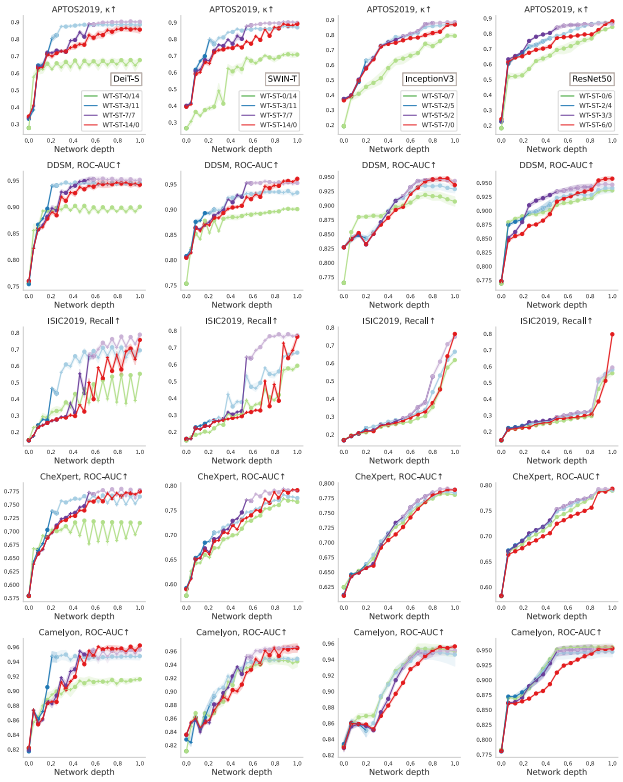

Figure 13. *Predictive performance of features at different depths using k-NN evaluation.* k-NN evaluation performance at different depths for models initialized with varying WT fractions, reported for each dataset (rows) and model type (columns). Overall the k-NN performance increases monotonically with depth for all models and datasets. However, relative performance gains from layer to layer exhibit different patterns. CNNs improve progressively, while ViTs increase rapidly in the beginning and then they reach a plateau. This plateau is observed to appear in association with the first ST-initialized layer in the WT-ST-n/m experiments.

similarity scores reported in these experiments follow the procedure described in [31]. For each setting the values are calculated by measuring the similarity over the full test set, in batches of 128. This is done for all five runs of each setting, and we report the mean similarity score averaged over all runs. The intermediate layers of the models that were used for calculating similarities could be seen in Table 2 and Table 3 in the Appendix and the results can be found in Figure 3 in the main text and Figures 9, 10, 11 and 12 in Appendix.

B. k-NN evaluation

We use k-NN evaluation to investigate the discriminative power at different layers throughout the network.. The evaluation is performed by comparing the similarity, in the feature space, of samples from the training and the test set. In particular, we use cosine similarity as the means to calcu-

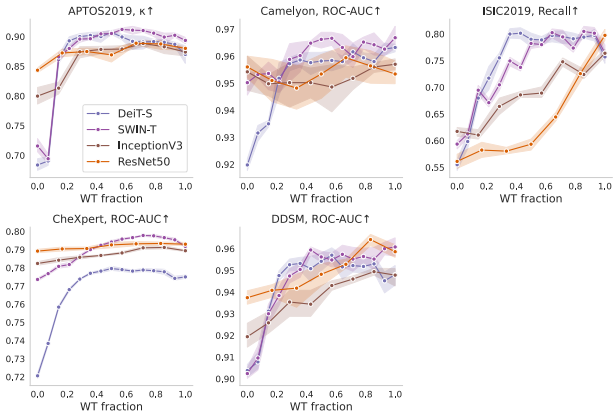

Figure 14. *Maximum k-nn predictive performance of intermediate features for different WT-ST-n/m initialization schemes*. Similar to Figure 13 relative gains exhibit different patterns of improvement. Once again ViTs appear to improve quickly early on and then plateau. However, this plateau has a negative slope in some cases, suggesting the presence of strong biases in the high-level features, possibly inherited from the pre-training task.

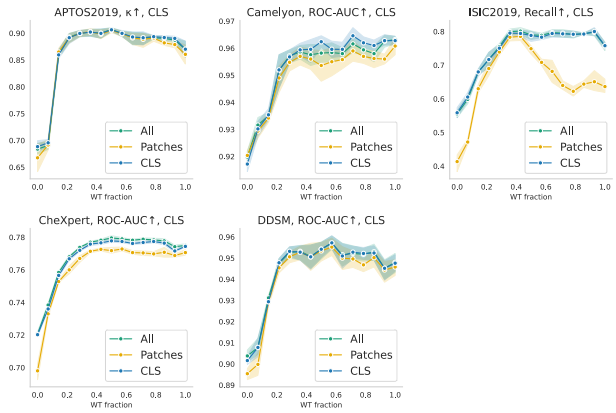

Figure 15. *Maximum k-nn predictive performance of intermediate features for different WT-ST-n/m initialization schemes when using different feature types from* DEIT-S *for evaluation*. Maximum k-nn evaluation score achieved at any depth for corresponding WT-ST-n/m initialization fraction, for DEIT-S *(1)* using only the `cls` token's activations, *(2)* using activations from the spatial tokens, *(3)* concatenating *(1)* and *(2)*. The different feature embeddings seem to exhibit similar trends , but the `cls` token often seem to outperform the patch embedding.

late the distance between different data-points. Then, labels are assigned to the query data-point, from the test set, by considering its k nearest neighbors from the training set. Throughout this work we use $k = 200$. The layers used to extract the embeddings are listed in Table 2 for CNNs and Table 3 for ViTs. The results of the k-NN evaluation experiments can be found in Figure 6 in the main text and Figures 13, 14 and 15 in the Appendix.

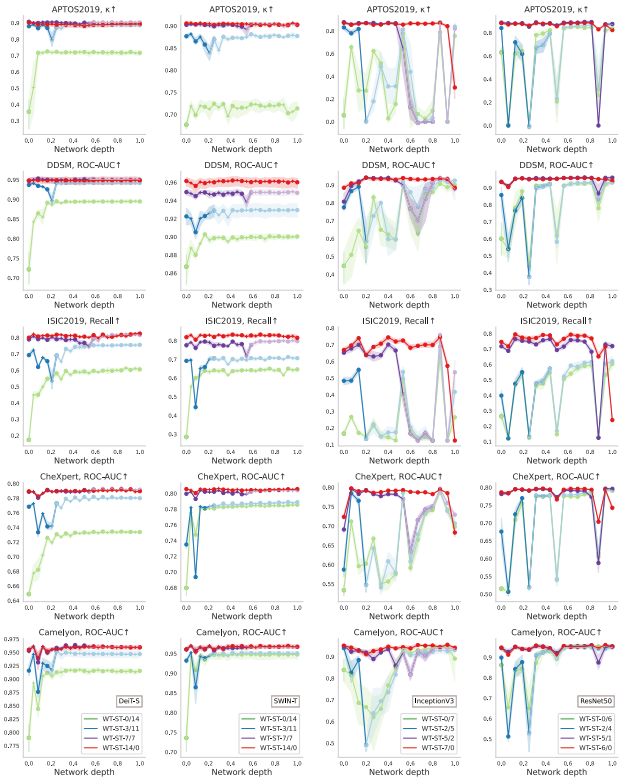

Figure 16. *Resilience of trained layers to change*. We report the performance when reverting a fine-tuned layer back to its original state. This is done using one layer at a time, for each dataset (rows) and model types (columns) for four different WT-ST-n/m initialization strategies. The results show that layers with low robustness underwent critical changes during fine-tuning. In ViTs, critical layers often appear at the transition between WT and ST. CNNs on the other hand, exhibit poor robustness at the final layers responsible for classification, and also periodically within the network at critical layers.

C. Re-initialization robustness

In the layer re-initialization experiments, we consider the impact of reverting individual layer of the models to their initial state. In detail, we initialize models with different WT-ST-n/m schemes. Then, after fine-tuning, we reinitialize a single layer at a time to its original state, while keeping all the other layers unchanged. Finally, we evaluate the model on the test set and measure the drop in predictive performance.

For DEITs and SWIN, the intermediate modules include the patchifier, the self-attention layers of each block separately. For RESNET50, the modules include the first convolutional layer and the residual blocks of each stage. For INCEPTION the modules consist of all the initial convolutional blocks, and all the individual inception modules. A detailed description of the layers that were used can be seen in Table 2 and Table 3 in Appendix. The results of these ex-

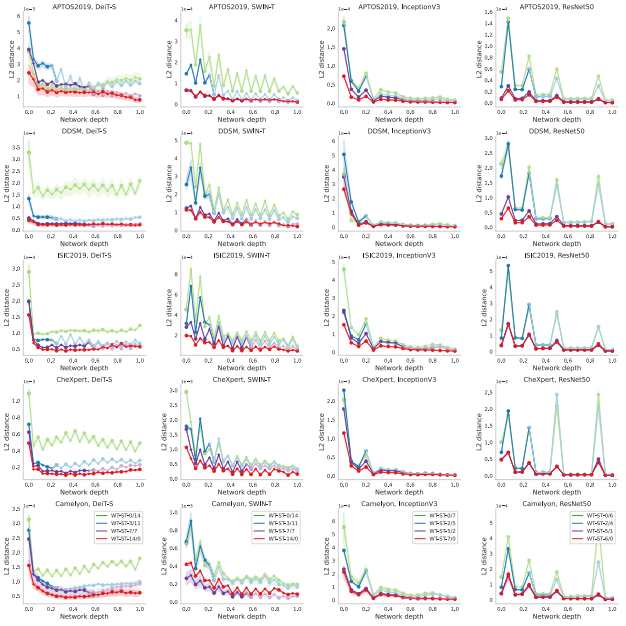

Figure 17. *ℓ_2 distance of the weights.* We report the the mean ℓ_2 distances between the initial and trained weights for different models with different initialization schemes. A large distance indicates that the corresponding layer has changed significantly during training.

periments can be seen in Figure 5 in the main text for DEIT and Figure 16 for other model types.

D. ℓ_2 distance

In order to understand the extent that model's weights change during training, we calculate the ℓ_2 norm of the weights before and after fine-tuning. In practice, for each layer, we calculate the ℓ_2 distances between the original and fine-tuned weights and then we divide this value by the number of the weights in the layer. The details of the layers used for each model can be seen in Table 2 and Table 3 in the Appendix. Figure 17 shows the the results for each model and dataset individually, and Figure 4 in the main text shows the distances averaged over all the datasets for each model.

E. Mean attended distance

To understand the type of features that emerge at different layer depths as a function of the initialization strategy WT-ST for DEITs, we calculate the mean attended distance per layer. That is, for each of the DEIT-S' attention heads we calculate the mean of the element-wise multiplication of each query token's attention and its distance from the other tokens, similarly to [11]. Then, we average the calculated distances per layer for all of the WT-ST initialization schemes. In Figure 18 in the Appendix, we report the

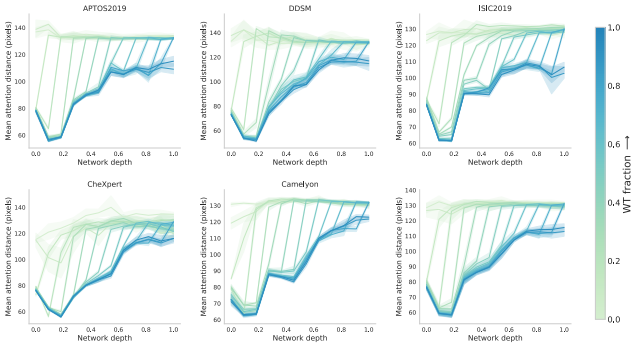

Figure 18. *Mean attended distance for different initializations.* We report the mean attended distance of the fine-tuned DEIT-S model for all datasets using different WT-ST initializations (WT fraction from 0 to 1, where 0 = ST and 1 = WT). The bottom-right figure shows the mean attended distance, averaged over all datasets. Evidently, in the absence of WT layers the attention is mainly global, whilst the IMAGENET pre-trained weights introduce a mixture of local and global attention that the network cannot learn on its own.

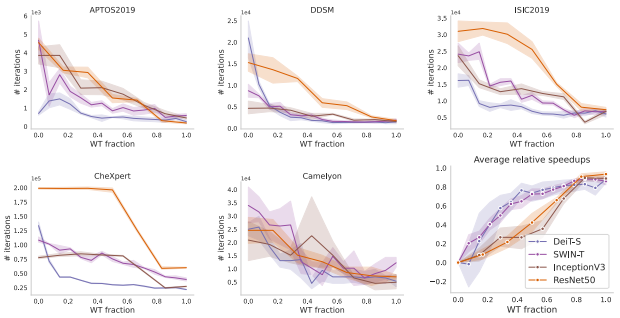

Figure 19. *Convergence speed as a function of WT fraction for different models and datasets.* We report the number of iterations it takes for each model to converge for different WT fractions for each individual dataset. The bottom-right figure shows the relative speedups averaged over all datasets. Evidently, the convergence speed monotonically increases with the number of WT layers for all datasets and models.

mean attended distance per layer for all datasets and the average attended distance over all datasets. The results clearly show that (1) the ST initialization results in global attention throughout the network (2) after the critical layers the attention is mainly global (3) the WT layers introduce a mixture of local and global features. This suggests that the WT layers are important for a mixture of local and global features that the model is incapable of learning on its own – due to the small data size.

F. Model convergence

We investigate how the convergence behavior of the models change as we transfer more layers. Figure 19 in the Appendix and Figure 8 in the main text show the number of iterations needed for each model to reach its best vali-

Initialization for Inception		Features of Inception	Initialization for ResNets		Features of ResNet50
WT-ST-n/m	**Module**	**Intermediate layers**	**WT-ST-n/m**	**Module**	**Intermediate layers**
WT-ST-1/6	Layer 1	Conv2d	WT-ST-1/5	Layer 1	Conv2d
WT-ST-2/5	Layer 2	Layer 2, Conv2d Layer 2, Conv2d	WT-ST-2/4	Norm layer (BN)	
WT-ST-3/4 WT-ST-4/3	Layer 3 Layer 4	Layer 3, Conv2d Layer 4, Conv2d	WT-ST-3/3	Stage 1	Stage 1, Block 1 Stage 1, Block 2 Stage 1, Block 3
WT-ST-5/2	Stage 1	Stage 1, Block 1 – Type A Stage 1, Block 3 – Type A Stage 1, Block 3 – Type A	WT-ST-4/2	Stage 2	Stage 2, Block 1 Stage 2, Block 2 Stage 2, Block 3 Stage 2, Block 4
WT-ST-6/1	Stage 2	Stage 2, Block 1 – Type B Stage 2, Block 2 – Type C Stage 2, Block 3 – Type C Stage 2, Block 4 – Type C Stage 2, Block 5 – Type C	WT-ST-5/1	Stage 3	Stage 3, Block 1 Stage 3, Block 2 Stage 3, Block 3 Stage 3, Block 4 Stage 3, Block 5 Stage 3, Block 6
WT-ST-7/0	Stage 3	Stage 3, Block 1 – Type D Stage 3, Block 2 – Type E Stage 3, Block 3 – Type E	WT-ST-6/0	Stage 4	Stage 4, Block 1 Stage 4, Block 2 Stage 4, Block 3

Table 2. *Implementation details for the CNN models*. **(Left)** Modules and layers used for INCEPTION. **(Right)** Modules and layers used for RESNETs. The first column of each side shows the initialization type that we used. The module that corresponds to each initialization scheme (second column from each side) is the last module that we initialized with WT. The modules after that were initialized with ST. The third row from each side shows the layers we used for the k-NN, ℓ_2, re-initialization and representation similarity experiments.

Initialization for DeiT		Features of DeiT-S	Initialization for SWIN-T		Features of SWIN-T
WT-ST-n/m	**Module**	**Intermediate layers**	**WT-ST-n/m**	**Module**	**Intermediate layers**
WT-ST-1/13	Patchifier	Conv2d	WT-ST-1/13	Patchifier	Conv2d
WT-ST-2/12	Block 1	Block 1 – Attention Block 1 – MLP	WT-ST-2/12	Stage 1, Block 1	Stage 1, Block 1 – Attention Stage 1, Block 1 – MLP
WT-ST-3/11	Block 2	Block 2 – Attention Block 2 – MLP	WT-ST-3/11	Stage 1, Block 2	Stage 1, Block 2 – Attention Stage 1, Block 2 – MLP
WT-ST-4/10	Block 3	Block 3 – Attention Block 3 – MLP	WT-ST-4/10	Stage 2, Block 1	Stage 2, Block 1 – Attention Stage 2, Block 1 – MLP
WT-ST-5/9	Block 4	Block 4 – Attention Block 4 – MLP	WT-ST-5/9	Stage 2, Block 2	Stage 2, Block 2 – Attention Stage 2, Block 2 – MLP
WT-ST-6/8	Block 5	Block 5 – Attention Block 5 – MLP	WT-ST-6/8	Stage 3, Block 1	Stage 3, Block 1 – Attention Stage 3, Block 1 – MLP
WT-ST-7/7	Block 6	Block 6 – Attention Block 6 – MLP	WT-ST-7/7	Stage 3, Block 2	Stage 3, Block 2 – Attention Stage 3, Block 2 – MLP
WT-ST-8/6	Block 7	Block 7 – Attention Block 7 – MLP	WT-ST-8/6	Stage 3, Block 3	Stage 3, Block 3 – Attention Stage 3, Block 3 – MLP
WT-ST-9/5	Block 8	Block 8 – Attention Block 8 – MLP	WT-ST-9/5	Stage 3, Block 4	Stage 3, Block 4 – Attention Stage 3, Block 4 – MLP
WT-ST-10/4	Block 9	Block 9 – Attention Block 9 – MLP	WT-ST-10/4	Stage 3, Block 5	Stage 3, Block 5 – Attention Stage 3, Block 5 – MLP
WT-ST-11/3	Block 10	Block 10 – Attention Block 10 – MLP	WT-ST-11/3	Stage 3, Block 6	Stage 3, Block 6 – Attention Stage 3, Block 6 – MLP
WT-ST-12/2	Block 11	Block 11 – Attention Block 11 – MLP	WT-ST-12/2	Stage 4, Block 1	Stage 4, Block 1 – Attention Stage 4, Block 1 – MLP
WT-ST-13/1	Block 12	Block 12 – Attention Block 12 – MLP	WT-ST-13/1	Stage 4, Block 2	Stage 4, Block 2 – Attention Stage 4, Block 2 – MLP
WT-ST-14/0	Layer norm		WT-ST-14/0	Layer norm	

Table 3. *Implementation details for the ViT models*. **(Left)** Modules and layers used for DEITs. **(Right)** Modules and layers used for SWIN-T. The first column of each side shows the initialization type that we used. The module that corresponds to each initialization scheme (second column from each side) is the last module that we initialized with WT. The modules after that were initialized with ST. The third row from each side shows the layers we used for the k-NN, ℓ_2, re-initialization and representation similarity experiments.

dation performance. As a general trend for all models, the higher the WT fraction is, the faster the models converge. Interestingly, for vision transformers, transferring the first few blocks dramatically increases the convergence speed, while transferring further blocks slightly speeds up training. CNNs however, follow a different trend, where transferring more layers improves the convergence speed at a roughly linear rate.

G. Model capacity

We investigate the impact that the model's capacity has on transfer learning for DEITs [39] and RESNETs [16]. To this end, we consider 3 different capacities for each architecture, which are comparable in the number of parameters and compute time. For the RESNET family, we considered RESNET18, RESNET50, and RESNET152. For the DEIT family, we chose DEIT-T, DEIT-S, and DEIT-B. For each model capacity, we carry out the same WT-ST-n/m experiments. That is, we initialize each model with different WT-ST-n/m initialization schemes and then we fine-tune them on the target task. The training strategy follows exactly the details mentioned in Section 2. Please refer to to Figure 8 and Section 3 for the results and discussion.

H. The WT-ST initialization schemes

Here, we provide additional details regarding the WT-ST-n/m initialization procedure and the modules we used to investigate where feature reuse occurs within the network. We transfer weights (WT) up to block n and we initialize the remaining m blocks using ST. In practice this means that, the first n modules use the exact ImageNet pretrained weights, while the weights of the next m modules are initialised with a Normal distribution $\mathcal{N}(\mu_i, {\sigma_i}^2)$, where μ_i and ${\sigma_i}^2$ are the mean and variance of the ith ImageNet pretrained weight.

Due to the architectural differences, we use a different selection of modules for each model. For DEITs and SWINs, we use the input layer (patchifier), each of the transformer blocks and the last normalization layer of the network. For INCEPTION we use the first four modules which include the layers that operate at the same scale and the inception modules that belong to the same stage. Finally, for RESNETs we include the input layer, the first normalization layer and the resnet blocks from each scale. The exact details for each WT-ST-n/m setting for RESNET50, RESNET18, RESNET152 and INCEPTION can be found in Table 2 in the Appendix. Similarly, the details for DEIT, DEIT-T, DEIT-B and SWIN-T are found in Table 3 in the Appendix. The results of these experiments are reported in Figure 2 and 7 in the main text.

I. The 5-layer DEIT-S model

As we showed in Figure 2 and Figure 6 of the main text, it appears that vision transformers benefit significantly from weight transfer in their initial-to-middle blocks, while trans-

ferring the weights in the later blocks seems to offer little or no benefit. In fact, transferring weights too deep into the network *may result in worse high-level features*, possibly due to biases learned during the pre-training task. Furthermore, we noticed from the layer-wise experiments that critical layers often appear at the transition between WT and ST. This begs the question: *Can we use a smaller* DEIT *model that has been initialized with weight transfer without compromising classification performance?*

To this end, we use a trimmed version of a DEIT-S model that has only five transformer blocks – effectively reducing the memory and computational requirements by a factor of 2. We initialize this model, denoted as DEIT-S-5b, with weight transfer from IMAGENET and we fine-tune it on the target datasets, using the settings described in Section 2. Surprisingly, our results in Table 4 showing no significant changes in classification performance. This supports the arguments that: *1)* the initial blocks of ViTs contribute the most to the overall performance of the model, *2)* feature reuse in the first layers is so strong for DEITs that can compensate for the lack of additional transformer blocks.

This finding might be of further interest for practitioners who work with limited computational and memory budgets. For example, in medical imaging, there is a need for light-weight models as the large image sizes that are encountered in practice prohibit the utilization of large models. However, further evaluation is needed to asses the extent to which these benefits are broadly applicable.

Model	**APTOS2019**, $\kappa \uparrow$ $n = 3{,}662$	**DDSM**, AUC $\uparrow$ $n = 10{,}239$	**ISIC2019**, Rec. $\uparrow$ $n = 25{,}333$	**CheXpert**, AUC $\uparrow$ $n = 224{,}316$	**Camelyon**, AUC $\uparrow$ $n = 327{,}680$
DeiT-S	0.894 ± 0.017	0.949 ± 0.011	0.824 ± 0.008	0.792 ± 0.001	0.962 ± 0.003
DeiT-S-5b	0.894 ± 0.005	0.951 ± 0.001	0.812 ± 0.016	0.792 ± 0.001	0.962 ± 0.004

Table 4. *A trimmed* DEIT-S *with only 5 blocks performs comparably to the full* DEIT-S *model.* We keep only the first 5 (out of 12) blocks of DEIT-S and discard the rest. Then, after WT initialization, we fine-tune the model with the same strategy detailed in Section 2. It can clearly be seen that the smaller model performs competitively to the full DEIT-S, even when more than half of the blocks are removed.